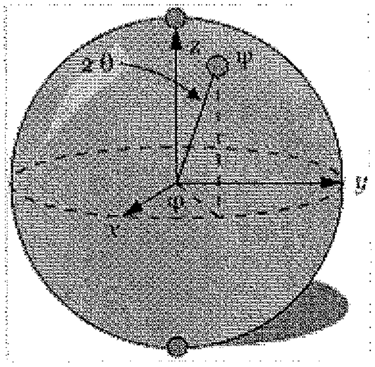
Qubit

From Wikipedia, the free encyclopedia

A qubit is not to be confused with a cubit, which is an ancient measure of length.



A **qubit** representation by a Bloch sphere.

A **quantum bit**, or **qubit** (sometimes *qbit*) ['kju.bɪt] *or* [k'bɪt] is a unit of quantum information. That information is described by a state vector in a two-level quantum mechanical system which is formally equivalent to a two-dimensional vector space over the complex numbers.

Benjamin Schumacher discovered a way of interpreting quantum states as information. He came up with a way of compressing the information in a state, and storing the information on a smaller number of states. This is now known as Schumacher compression. In the acknowledgments of his paper (Phys. Rev. A 51, 2738), Schumacher states that the term qubit was invented in jest, during his conversations with Bill Wootters.

Contents

Bit versus qubit

A bit is the base of computer information. Regardless of its physical representation, it is always read as either a 0 or a 1. An analogy to this is a light switch - the down position can represent 0 (normally equated to *off*) and the up position can represent 1 (normally equated to *on*).

A qubit has some similarities to a classical bit, but is overall very different. Like a bit, a qubit can have only two possible values - normally a 0 or a 1. The difference is that whereas a bit *must* be either 0 or 1, a qubit can be 0, 1, or a superposition of both.

Representation

The states a qubit may be measured in are known as basis states (or vectors). As is the tradition with any sort of quantum states, Dirac, or bra-ket notation is used to represent them.

This means that the two computational basis states are conventionally written as $|0\rangle$ and $|1\rangle$ (pronounced: 'ket 0' and 'ket 1').

Qubit states

A pure qubit state is a linear superposition of those two states. This means that the qubit can be represented as a linear combination of $|0\rangle$ and $|1\rangle$:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle,$$

where α and β are probability amplitudes and can in general both be complex numbers.

When we measure this qubit in the standard basis, the probability of outcome $|0\rangle$ is $|\alpha|^2$ and the probability that the outcome is $|1\rangle$ is $|\beta|^2$. Because the absolute squares of the amplitudes equate to probabilities, it follows that α and β must be constrained by the equation

$$|\alpha|^2 + |\beta|^2 = 1$$

simply because this ensures you must measure either one state or the other.

The state space of a single qubit register can be represented geometrically by the Bloch sphere. This is a two dimensional space which has an underlying geometry of the surface of a sphere. This essentially means that the single qubit register space has two local degrees of freedom. Represented on such a sphere, a classical bit could only lie on one of the poles. An n-qubit register space has $2^{n+1} - 2$ degrees of freedom. This is much larger than $2n$, which is what one would expect classically with no entanglement.

Entanglement

An important distinguishing feature between a qubit and a classical bit is that multiple qubits can exhibit quantum entanglement. Entanglement is a nonlocal property that allows a set of qubits to express higher correlation than is possible in classical systems. Take, for example, two entangled qubits in the Bell state

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle).$$

(Note that in this state, there are equal probabilities of measuring either $|00\rangle$ or $|11\rangle$.)

Imagine that these two entangled qubits are separated, with one each given to Alice and Bob. Alice makes a measurement of her qubit, obtaining - with equal probabilities - either $|0\rangle$ or $|1\rangle$. Because of the qubits' entanglement, Bob must now get the exact same measurement as Alice, i.e. if she measured a $|0\rangle$, Bob must measure the same, as $|00\rangle$ is the only state where Alice's qubit is a $|0\rangle$.

Entanglement also allows multiple states (such as are the Bell state mentioned above) to be acted on simultaneously, unlike classical bits that can only have one value at a time. Entanglement is a necessary ingredient of any quantum computation that cannot be done efficiently on a classical computer.

The use of entanglement in quantum computing has been referred to as "quantum parallelism", and offers a possible explanation for the power of quantum computing: because the state of the computer can be in a quantum superposition of many different classical computational paths, these paths can all proceed concurrently.

Quantum register

A number of entangled qubits taken together is a qubit register. Quantum computers perform calculations by manipulating qubits within a register.

Variations of the qubit

Similar to the qubit, a qutrit is a unit of quantum information in a 3-level quantum system. This is analogous to the unit of classical information trit. The term "**Qudit**" is used to denote a unit of quantum information in a d-level quantum system.

Physical representation

Any two-level system can be used as a qubit. Multilevel systems can be used as well, if they possess two states can be effectively decoupled from the rest (e.g., ground state and first excited state of a nonlinear oscillator). There are various proposals. Several physical implementations which approximate two-level systems to various degrees were successfully realized. Similarly to a classical bit where the state of a transistor in a processor, the magnetization of a surface in a hard disk and the presence of current in a cable can all be used to represent bits in the same computer, an eventual quantum computer is likely to use various combinations of qubits in its design.

This is an incomplete list of physical implementation of qubits:

Physical support	Name	Information support	"0"	"1"
Single photon (Fock states)	Polarization encoding	Polarization of light	Horizontal	Vertical
	Photon number	Photon number	Vacuum	Single photon state
	Time-bin encoding	Time of arrival	Early	Late
Coherent state of light	Squeezed light	Quadrature	Amplitude-squeezed state	Phase-squeezed state
Electrons	Electronic spin	Spin	Up	Down
	Electron number	Charge	No electron	One electron
Nucleus	Nuclear spin addressed through NMR	Spin	Up	Down
Optical lattices	Atomic spin	Spin	Up	Down
Josephson junction	Superconducting charge qubit	Charge	Uncharged superconducting island (Q=0)	Charged superconducting island (Q=2e, one extra Cooper pair)
	Superconducting flux qubit	Current	Clockwise current	Counterclockwise current
	Superconducting phase qubit	Energy	Ground state	First excited state
Singly-charged quantum dot pair	Electron localization	Charge	Electron on left dot	Electron on right dot

External links

- An update on qubits in the Jan 2005 (http://www.sciam.com/) issue of Scientific American
- An update on qubits in the Oct 2005 (http://www.sciam.com/article.cfm?chanID=sa006&colID=5&articleID=000D4372-A8A9-1330-A54583414B7F0000) issue of Scientific American
- The organization (http://www.qubit.org/) cofounded by one of the pioneers in quantum computation, David Deutsch

Quantum computing

Qubit | Quantum circuit | Quantum computer | Quantum cryptography | Quantum information | Quantum programming | Quantum teleportation | Quantum virtual machine | Timeline of quantum computing

Nuclear magnetic resonance (NMR) quantum computing

Liquid-state NMR QC | Solid-state NMR QC

Photonic computing

Nonlinear optics | Linear optics QC | Non-linear optics QC | Coherent state based QC

Trapped ion quantum computer

NIST-type ion-trap QC | Austria-type ion-trap QC

Semiconductor-based quantum computing

Kane QC | Loss-DiVincenzo QC

Superconducting quantum computing

Charge qubit | Flux qubit | Hybrid qubits

Retrieved from "http://en.wikipedia.org/wiki/Qubit"

Categories: Units of information | Quantum information science

- This page was last modified 08:41, 25 February 2007.

Grover's algorithm

From Wikipedia, the free encyclopedia
(Redirected from Grovers algorithm)

Grover's algorithm is a quantum algorithm for searching an unsorted database with N entries in $O(N^{1/2})$ time and using $O(\log N)$ storage space (see big O notation). It was invented by Lov Grover in 1996.

Classically, searching an unsorted database requires a linear search, which is $O(N)$ in time. Grover's algorithm, which takes $O(N^{1/2})$ time, is the fastest possible quantum algorithm for searching an unsorted database. It provides "only" a quadratic speedup, unlike other quantum algorithms, which may provide exponential speedup over their classical counterparts. However, even quadratic speedup is considerable when N is large.

Like many quantum computer algorithms, Grover's algorithm is probabilistic in the sense that it gives the correct answer with high probability. The probability of failure can be decreased by repeating the algorithm. (An example of a deterministic quantum algorithm is the Deutsch-Jozsa algorithm, which always produces the correct answer with probability one.)

Contents

Applications

Although the purpose of Grover's algorithm is usually described as "searching a database", it may be more accurate to describe it as "inverting a function". Roughly speaking, if we have a function $y=f(x)$ that can be evaluated on a quantum computer, this algorithm allows us to calculate x when given y. Inverting a function is related to the searching of a database because we could come up with a function that produces a particular value of y if x matches a desired entry in a database, and another value of y for other values of x.

Grover's algorithm can also be used for estimating the mean and median of a set of numbers, and for solving the collision problem. In addition, it can be used to solve NP-complete problems by performing exhaustive searches over the set of possible solutions. This would result in a considerable speedup over classical solutions, even though it does not provide the "holy grail" of a polynomial-time solution.

Below, we present the basic form of Grover's algorithm, which searches for a single matching entry. The algorithm can be further optimized if there is more than one matching entry and the number of matches is known beforehand.

Setup

Consider an unsorted database with N entries. The algorithm requires an N-dimensional state space H, which can be supplied by $\log_2 N$ qubits.

Let us number the database entries by 0, 1, ... (N-1). Choose an observable, Ω, acting on H, with N distinct eigenvalues whose values are all known. Each of the eigenstates of Ω encode one of the entries in the database, in a manner that we will describe. Denote the eigenstates (using bra-ket notation) as

$$\{|0\rangle, |1\rangle, \cdots, |N-1\rangle\}$$

and the corresponding eigenvalues by

$$\{\lambda_0, \lambda_1, \cdots, \lambda_{N-1}\}$$

We are provided with a unitary operator, U_ω, which acts as a subroutine that compares database entries according to some search criterion. The algorithm does not specify how this subroutine works, but it must be a *quantum* subroutine that works with superpositions of states. Furthermore, it must act specially on one of the eigenstates, |ω>, which corresponds to the database entry matching the search criterion. To be precise, we require U_ω to have the following effects:

$$\begin{aligned} U_\omega|\omega\rangle &= -|\omega\rangle \\ U_\omega|x\rangle &= |x\rangle \qquad \text{for all } x \neq \omega \end{aligned}$$

Our goal is to identify this eigenstate |ω>, or equivalently the eigenvalue ω, that U_ω acts specially upon.

Algorithm breakdown

The steps of Grover's algorithm are as follows:

1. Initialize the system to the state
$$|s\rangle = \frac{1}{\sqrt{N}} \sum_x |x\rangle$$
2. Perform the following "Grover iteration" *r(N)* times. The function *r(N)* is described below.
    1. Apply the operator U_ω
    2. Apply the operator $U_s = 2\,|s\rangle\,\langle s| - I.$
3. Perform the measurement Ω. The measurement result will be λ_ω with probability approaching 1 for N>>1. From λ_ω, ω may be obtained.

Our initial state is

$$|s\rangle = \frac{1}{\sqrt{N}} \sum_x |x\rangle$$

Consider the plane spanned by |s> and |ω>. Let $|\omega^\times\rangle$ be a ket in this plane perpendicular to |ω>. Since |ω> is one of the basis vectors, the overlap is

$$\langle\omega|s\rangle = \frac{1}{\sqrt{N}}$$

In geometric terms, there is an angle (π/2 - θ) between |ω> and |s>, where θ is given by:

$$\begin{aligned} \cos\left(\frac{\pi}{2} - \theta\right) &= \frac{1}{\sqrt{N}} \\ \sin\theta &= \frac{1}{\sqrt{N}} \end{aligned}$$

The operator U_ω is a reflection at the hyperplane orthogonal to |ω>; for vectors in the plane spanned by |s> and |ω>, it acts as a reflection at the line through $|\omega^\times\rangle$. The operator U_s is a reflection at the line through |s>. Therefore, the state vector remains in the plane spanned by |s> and |ω> after each application of U_s and after each application of U_ω, and it is straightforward to check that the operator U_sU_ω of each Grover iteration step rotates the state vector by an angle of 2θ toward |ω>.

We need to stop when the state vector passes close to |ω>; after this, subsequent iterations rotate the state vector *away* from |ω>, reducing the probability of obtaining the correct answer. The number of times to iterate is given by *r*. In order to align the state vector exactly with |ω>, we need:

$$\frac{\pi}{2} - \theta = 2\theta r$$
$$r = \frac{(\frac{\pi}{\theta} - 2)}{4}$$

However, r must be an integer, so generally we can only set r to be the integer closest to $(\pi/\theta - 2)/4$. The angle between $|\omega\rangle$ and the final state vector is $O(\theta)$, so the probability of obtaining the wrong answer is $O(1 - \cos^2\theta) = O(\sin^2\theta)$.

For $N>>1$, $\theta \approx N^{-1/2}$, so

$$r \longrightarrow \frac{\pi\sqrt{N}}{4}$$

Furthermore, the probability of obtaining the wrong answer becomes $O(1/N)$, which goes to zero for large N.

Extensions

If, instead of 1 matching entry, there are k matching entries, the same algorithm works but the number of iterations must be $\pi(N/k)^{1/2}/4$ instead of $\pi N^{1/2}/4$. There are several ways to handle the case if k is unknown. For example, one could run Grover's algorithm several times, with

$$\pi\frac{N^{1/2}}{4}, \pi\frac{(N/2)^{1/2}}{4}, \pi\frac{(N/4)^{1/2}}{4}, \ldots$$

iterations. For any k, one of iterations will find a matching entry with a sufficiently high probability. The total number of iterations is at most

$$\pi\frac{N^{1/2}}{4}\left(1 + \frac{1}{\sqrt{2}} + \frac{1}{2} + \cdots\right)$$

which is still $O(N^{1/2})$.

Optimality

It is known that Grover's algorithm is optimal. That is, any algorithm that accesses the database only by using the operator U_ω must apply U_ω at least as many times as Grover's algorithm (Bennett et al., 1997). This result is important in understanding the limits of quantum computation. If the Grover's search problem was solvable with $\log^c N$ applications of U_ω, that would imply that NP is contained in BQP, by transforming problems in NP into Grover-type search problems. The optimality of Grover's algorithm suggests (but does not prove) that NP is not contained in BQP.

The number of iterations for k matching entries, $\pi(N/k)^{1/2}/4$, is also optimal.

References

1. Grover L.K.: *A fast quantum mechanical algorithm for database search (http://arxiv.org/abs/quant-ph/9605043)*, Proceedings, 28th Annual ACM Symposium on the Theory of Computing, (May 1996) p. 212
2. Grover L.K.: *From Schrödinger's equation to quantum search algorithm (http://arxiv.org/abs/quant-ph/0109116)*, American Journal of Physics, 69(7): 769-777, 2001. Pedagogical review of the algorithm and its history.
3. http://www.bell-labs.com/user/feature/archives/lkgrover/

4. Bennett C.H., Bernstein E., Brassard G., Vazirani U., *The strengths and weaknesses of quantum computation*. SIAM Journal on Computing 26(5): 1510-1523 (1997). Shows the optimality of Grover's algorithm.

- Quantum search algorithms on arxiv.org (http://xstructure.inr.ac.ru/x-bin/theme2.py?arxiv=quant-ph&level=1&index1=1437)

Retrieved from "http://en.wikipedia.org/wiki/Grover%27s_algorithm"

Categories: Quantum algorithms | Search algorithms

Shor's algorithm

From Wikipedia, the free encyclopedia
(Redirected from Shors algorithm)

Shor's algorithm is a quantum algorithm for factoring a number N in O((log N)^3) time and O(log N) space, named after Peter Shor.

The algorithm is significant because it implies that RSA, a popular public-key cryptography method, might be easily broken, given a sufficiently large quantum computer. RSA uses a public key N which is the product of two large prime numbers. One way to crack RSA encryption is by factoring N, but with classical algorithms, factoring becomes increasingly time-consuming as N grows large; more specifically, no classical algorithm is known that can factor in time O((log N)^k) for any k. By contrast, Shor's algorithm can crack RSA in polynomial time.

Like many quantum computer algorithms, Shor's algorithm is probabilistic: it gives the correct answer with high probability, and the probability of failure can be decreased by repeating the algorithm. However, since a proposed answer (in particular primality) is polynomial time verifiable, the algorithm can be modified to work in expected polynomial time with zero error.

Shor's algorithm was discovered in 1994 by Peter Shor, but the classical part was known before, it is credited to G. L. Miller. Seven years later, in 2001, it was demonstrated by a group at IBM, which factored 15 into 3 and 5, using a quantum computer with 7 qubits.

Contents

Procedure

The problem we are trying to solve is that, given an integer N, we try to find another integer p between 1 and N that divides N.

Shor's algorithm consists of two parts:

1. A reduction of the factoring problem to the problem of order-finding, which can be done on a classical computer.
2. A quantum algorithm to solve the order-finding problem.

Classical part

1. Pick a random number $a < N$
2. Compute gcd(a, N). This may be done using the Euclidean algorithm.
3. If gcd(a, N) ≠ 1, then there is a nontrivial factor of N, so we are done.
4. Otherwise, use the period-finding subroutine (below) to find r, the period of the following function:
   $$f(x) = a^x \bmod N,$$
   i.e. the smallest integer r for which $f(x + r) = f(x)$.
5. If r is odd, go back to step 1.
6. If $a^{r/2} \equiv -1 \pmod N$, go back to step 1.
7. The factors of N are gcd($a^{r/2} \pm 1$, N). We are done.

Quantum part: Period-finding subroutine

1. Start with a pair of input and output qubit registers with $\log_2 N$ qubits each, and initialize them to
$$N^{-1/2}\sum_x |x\rangle\,|0\rangle$$
where x runs from 0 to N - 1.
2. Construct $f(x)$ as a quantum function and apply it to the above state, to obtain
$$N^{-1/2}\sum_x |x\rangle\,|f(x)\rangle$$
3. Apply the inverse quantum Fourier transform on the input register. The inverse quantum Fourier transform on N points is defined by:
$$U_{QFT}\,|x\rangle = N^{-1/2}\sum_y e^{-2\pi ixy/N}\,|y\rangle$$
This leaves us in the following state:
$$N^{-1}\sum_x\sum_y e^{-2\pi ixy/N}\,|y\rangle\,|f(x)\rangle$$
4. Perform a measurement. We obtain some outcome y in the input register and $f(x_0)$ in the output register. Since f is periodic, the probability of measuring some pair y and $f(x_0)$ is given by
$$\left|N^{-1}\sum_{x:\,f(x)=f(x_0)} e^{-2\pi ixy/N}\right|^2 = N^{-2}\left|\sum_b e^{-2\pi i(x_0+rb)y/N}\right|^2$$
Analysis now shows that this probability is higher, the closer yr/N is to an integer.
5. Turn y/N into an irreducible fraction, and extract the denominator r', which is a candidate for r.
6. Check if $f(x) = f(x + r')$. If so, we are done.
7. Otherwise, obtain more candidates for r by using values near y, or multiples of r'. If any candidate works, we are done.
8. Otherwise, go back to step 1 of the subroutine.

Explanation of the algorithm

The algorithm is composed of two parts. The first part of the algorithm turns the factoring problem into the problem of finding the period of a function, and may be implemented classically. The second part finds the period using the inverse quantum Fourier transform, and is responsible for the quantum speedup.

I. Obtaining factors from period

The integers less than N and coprime with N form a finite group under multiplication modulo N, which is typically denoted $(\mathbf{Z}/N\mathbf{Z})^\times$. By the end of step 3, we have an integer a in this group. Since the group is finite, a must have a finite order r, the smallest positive integer such that

$$a^r \equiv 1 \bmod N.$$

Therefore, $N \mid (a^r - 1)$. Suppose we are able to obtain r, and it is even. Then

$$a^r - 1 = (a^{r/2} - 1)(a^{r/2} + 1) \equiv 0 \bmod N$$
$$\Rightarrow N \mid (a^{r/2} - 1)(a^{r/2} + 1).$$

r is the *smallest* positive integer such that $a^r \equiv 1$, so N cannot divide $(a^{r/2} - 1)$. If N also does not divide $(a^{r/2} + 1)$, then N must have a nontrivial common factor with each of $(a^{r/2} - 1)$ and $(a^{r/2} + 1)$.

Proof: For simplicity, denote $(a^{r/2} - 1)$ and $(a^{r/2} + 1)$ by u and v respectively. $N \mid uv$, so $kN = uv$ for some integer k. Suppose $\gcd(u, N) = 1$; then $mu + nN = 1$ for some integers m and n (this is a property of the greatest common divisor.) Multiplying both sides by v, we find that $mkN + nvN = v$, so $N \mid v$. By contradiction, $\gcd(u, N) \neq 1$. By a similar argument, $\gcd(v, N) \neq 1$.

This supplies us with a factorization of N. If N is the product of two primes, this is the *only* possible factorization.

II. Finding the period

Shor's period-finding algorithm relies heavily on the ability of a quantum computer to be in many states simultaneously. Physicists call this behaviour a "superposition" of states. To compute the period of a function *f*, we evaluate the function at all points simultaneously.

Quantum physics does not allow us to access all this information directly, though. A measurement will yield only one of all possible values, destroying all others. Therefore we have to carefully transform the superposition to another state that will return the correct answer with high probability. This is achieved by the inverse quantum Fourier transform.

Shor thus had to solve three "implementation" problems. All of them had to be implemented "fast", which means that they can be implemented with a number of quantum gates that is polynomial in $\log N$.

1. Create a superposition of states. This can be done by applying Hadamard gates to all qubits in the input register. Another approach would be to use the quantum Fourier transform (see below).
2. Implement the function *f* as a quantum transform. To achieve this, Shor used repeated squaring for his modular exponentiation transformation. It is important to note that this step is more difficult to implement than the quantum Fourier transform, in that it requires ancillary qubits and substantially more gates to accomplish.
3. Perform an inverse quantum Fourier transform. By using controlled rotation gates and Hadamard gates Shor designed a circuit for the quantum Fourier transform that uses just $O((\log N)^2)$ gates.

After all these transformations a measurement will yield an approximation to the period r. For simplicity assume that there is a y such that yr/N is an integer. Then the probability to measure y is 1. To see that we notice that then

$$e^{-2\pi i b y r / N} = 1$$

for all integers b. Therefore the sum whose square gives us the probability to measure y will be N/r since b takes roughly N/r values and thus the probability is $1 / r^2$. There are r y such that yr/N is an integer and also r possibilities for $f(x_0)$, so the probabilities sum to 1.

Note: another way to explain Shor's algorithm is by noting that it is just the quantum phase estimation algorithm in disguise.

Modifications to Shor's Algorithm

There have been many modifications to Shor's algorithm. For example, whereas, an order of twenty to thirty runs are required on a quantum computer in the case of Shor's original algorithm, and with some of the other modifications, in the case of the modification done by David McAnally at the University of Queensland an order of only four to eight runs on the quantum computer is required.

References

Preprint of the original paper by Peter Shor:

- *Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer*, Peter W. Shor (http://www.arxiv.org/abs/quant-ph/9508027)

A general textbook on quantum computing:

- *Quantum Computation and Quantum Information*, Michael A. Nielsen, Isaac L. Chuang, Cambridge University Press, 2000

An implementation of Shor's Algorithm that factorizes the number 15:

- *Efficient Networks for Quantum Factoring*, David Beckman, Amalavoyal N. Chari, Srikrishna Devabhaktuni, and John Preskill, Phys. Rev. A 54, 1034–1063 (1996).

- Experimental realization of Shor's quantum factoring algorithm using nuclear magnetic resonance (http://cryptome.org/shor-nature.htm)

Lieven M. K. Vandersypen, Matthias Steffen, Gregory Breyta, Costantino S. Yannoni, Mark H. Sherwood & Isaac L. Chuang

- Implementing Shor's algorithm on Josephson Charge Qubits (http://arxiv.org/PS_cache/quant-ph/pdf/0308/0308171.pdf)

Juha J. Vartiainen, Antti O. Niskanen, Mikio Nakahara, Martti M. Salomaa

- Implementation of Shor's Algorithm on a Linear Nearest Neighbour Qubit Array (http://arxiv.org/PS_cache/quant-ph/pdf/0402/0402196.pdf)

Austin G. Fowler, Simon J. Devitt, Lloyd C. L. Hollenberg

External links

- "Explanation for the man in the street" by Scott Aaronsen (http://scottaaronson.com/blog/?p=208), "approved" by Peter Shor
- A demonstration of Shor's algorithm in PHP (http://pdivos.mobstop.com/shor/)

Retrieved from "http://en.wikipedia.org/wiki/Shor%27s_algorithm"

Categories: Quantum algorithms | Integer factorization algorithms | Quantum information science

Search Site

Algebra
Applied Mathematics
Calculus and Analysis
Discrete Mathematics
Foundations of Mathematics
Geometry
History and Terminology
Number Theory
Probability and Statistics
Recreational Mathematics
Topology

Alphabetical Index
Interactive Entries
Random Entry
New in *MathWorld*

MathWorld Classroom

About *MathWorld*
Send a Message to the Team

Order book from Amazon

Last updated:
12,645 entries
Wed Feb 14 2007

Created, developed, and nurtured by Eric Weisstein at Wolfram Research

History and Terminology > Notation
Calculus and Analysis > Series > Asymptotic Series

Landau Symbols

COMMENT On this Page

CONTRIBUTE TO THIS ENTRY!

Let n be an integer variable which tends to infinity and let x be a continuous variable tending to some limit. Also, let $\phi(n)$ or $\phi(x)$ be a positive function and $f(n)$ or $f(x)$ any function. Then the symbols $O(x)$ (sometimes called "big-O") and $o(x)$ (sometimes called "little-o") are known as the Landau symbols and defined as follows.

1. $f = O(\phi)$ means that $|f| < A\phi$ for some constant A and all values of n and x,

2. $f = o(\phi)$ means that $f/\phi \to 0$

(Hardy and Wright 1979, pp. 7-8).

Historically speaking, the symbol $O(x)$ first appeared in the second volume of Bachmann's treatise on number theory (Bachmann 1894), and Landau obtained this notation in Bachmann's book (Landau 1909, p. 883; Derbyshire 2004, p. 238). However, the symbol $o(x)$ did indeed originate with Landau (1909) in place of the earlier notation $\{x\}$ (Narkiewicz 2000, p. XI).

SEE ALSO: Asymptotic Notation.
[Pages Linking Here]

REFERENCES:

Bachmann, P. *Analytische Zahlentheorie, Bd. 2: Die Analytische Zahlentheorie.* Leipzig, Germany: Teubner, 1894.

de Bruijn, N. G. *Asymptotic Methods in Analysis.* New York: Dover, pp. 3-10, 1981.

Derbyshire, J. *Prime Obsession: Bernhard Riemann and the Greatest Unsolved Problem in Mathematics.* New York: Penguin, 2004.

Hardy, G. H. and Wright, E. M. "Some Notations." §1.6 in *An Introduction to the Theory of Numbers, 5th ed.* Oxford, England: Clarendon Press, pp. 7-8, 1979.

Havil, J. "Big Oh Notation." Appendix B in *Gamma: Exploring Euler's Constant.* Princeton, NJ: Princeton University Press, p. 219, 2003.

Miller, J. "Earliest Uses of Symbols of Number Theory." http://members.aol.com/jeff570/nth.html.

Landau, E. *Handbuch der Lehre von der Verteilung der Primzahlen.* Leipzig, Germany: Teubner, 1909. Reprinted by New York: Chelsea, 1953.

Narkiewicz, W. *The Development of Prime Number Theory: From Euclid to Hardy and Littlewood.* New York: Springer-Verlag, 2000.

LAST MODIFIED: July 17, 2002

CITE THIS AS:

Weisstein, Eric W. "Landau Symbols." From *MathWorld*--A Wolfram Web Resource. http://mathworld.wolfram.com/LandauSymbols.html